DØ note 3191
Revision A

April 11, 1997

D0gstar
DØ GEANT Simulation of the Total Apparatus Response

Yuri Fisyak, John Womersley

Abstract

The purpose of this simulation package is to offer a central facility for Monte-Carlo studies of the DØ detector with different configurations. It allows a non-specialist to have a full GEANT simulation of the apparatus with a simple interface, while also providing the specialist with a full description of all the subdetectors. The package allows the user to run separately each of the different phases of detector simulation: event generation (**d0gen**), particle tracking in the detector with hit generation (**d0sim**), pile-up, signal and noise simulation (**d0raw**). It also includes simple reconstruction (**d0rec**) and analysis (**d0ana**) tools.

In order to provide a smooth transition from GEANT3 to GEANT4, we have made strenuous efforts to stay within the GEANT framework, especially in the treatment of hits. This Users' Guide contains general information on how to install, how to run and control the DØ simulation package. The guide and the reference manual will be constantly updated. Sections of the document are maintained by the various people responsible. The Users' Guide is available on the web via the DØ software pages. One can also get a postscript copy of the document via

The DØ Experiment → The DØ Upgrade → Offline Software → Monte Carlo → DØgstar

or directly

`http://d0sgi0.fnal.gov/~d0upgrad/d0_private/software/montecarlo/d0gstar/tex/manual.ps`

Contents

List of Figures

List of Tables

Part 1

User's Guide

1.1 Introduction

1.1.1 General

The **D0gstar** simulation package relies heavily on the CERN package GEANT[1]. Features and properties of GEANT are used for simulating particle behaviour in the detector (detector simulation), for saving the hits, and for reconstruction and analysis (if desired).

D0gstar simulates the performance of the DØ detector for Run II at the Tevatron. The Run I detector geometry is also available within **D0gstar** but digitization is not handled at present.

In this document we will mainly pay attention to simulation in the "Global" context but we have tried to create a framework which includes also the requirements for "FAST" and "MIXTURE" level simulations.

1.1.2 Basic framework

The basic differences between the old UPG_GEANT and **D0gstar** are as follows:

- The software is primarily organized on the basis of functionality and only secondarily on the basis of subdetectors:
  - All geometry definitions are in a single file, **detc** ;
  - All constants to build the geometry are in data file(s);
  - Hit definitions for all tracking detectors – the Central (Inner) tracker and the Muon system (which is considered to be the Outer tracker) – are unified and the differences among the specific detectors appear only in the digitization and cluster reconstruction stages. The format of reconstructed hits is the same for all tracking detectors in DØ.
- The version selection mechanism has been changed.
- Naming conventions have been revised. The numbering convention is obsolete; no numbers for rotation matrices or materials need be used as access to all these values is done via character string.
- Support for pileup, including that from preceding and following beam crossings.

Details of the new conventions are described in the following sections. Information on administration and maintenance of the **D0gstar** system is given in Appendix .1.

1.1.3 Goals of the full simulation

D0gstar is the detector simulation package for the DØ experiment. Among the **goals for full DØ simulation** are the following:

- to create an adequate computer model of the detector;
- to simulate the detector response up to the level of the data that will be obtained from the DØ DAQ system;
- to provide a framework for reconstructing events as a debugging tool.

At least three levels of detail must be provided for different subdetector simulation models:

- fast simulation – geometrical acceptance + parametrical model for detector responses
- the so-called "mixture" level simulation – dead materials, background estimation and reconstruction
- full simulation – detailed event simulation up to the DAQ level accounting for event pile-up effects.

It is important to note that a consistent simulation must be provided at each of these different levels. An essential component of the computer model of the detector is the "data base" consisting of the following elements:

- a description of the detector geometry
- the magnetic field map
- all material and tracking media definitions
- sensitive element definitions
- definitions of "hits" providing the information required for the digitization stage and event reconstruction
- the tools necessary to work with this data base

1.2 Computing environment

1.2.1 DØ machines

At present the **D0gstar** simulation system is supported on the following platforms at **Fermilab** :

- d0sgi0 and d0cha/d0chb (DØ sgi cluster)
- d0tng (DØ dedicated Open VMS cluster)
- fnalu (**Fermilab central Unix** cluster)

The reference machine for code maintenance is d0sgi. The most up-to-date version of the **D0gstar** package is always found there and versions on other platforms should be just copies from d0sgi.

1.2.2 Code maintenance

The source code is maintained using CVS [2]. The ASCII data files, such as GEANT detector definition constants and magnetic field map, are also stored in CVS in the format accessible by the ZEBRA [4] TZ package (i.e. TZ titles) [5].

The public CVS source files are kept in the directory $CVSROOT[1]. The breakdown of these files in the directories is as follows:

```
Repository: /cvsroot/d0cvs
   Module: d0gstar
           +---docs/          D0gstar documentation
           |   +---html/            (HTML files)
           |   +---tex/             (TeX, style and eps files)
           +---init/          set of environment variables initialization scripts
           +---mgr/           set of scripts for compilation, linking and run
           +---src/           top level directory for FORTRAN sources
           |   +---d0si/      Simulation steering
           |   +---detc/      Detector Constants
           |   +---util/      general Utilities
           |   +---gene/      Interface with PYTHIA, HERWIG, ISAJET
           |   +---sys/       header files with D0 and CERNLIB machine definitions
           |   +---trak/      central Tracker related codes
           |   +---calo/      Calorimeter       -"-
           |   +---muon/      Muon system       -"-
           |   +---user/      Reconstruction    -"-
           |   +---reco/      User specific     -"-
           |   +---gcal/      GEANT-GCALor interface
           +---d0db/          top level directory for the data
           |   +---detc/      set of tz-files with detector constant description
           |                  and rz-files with GEANT databases
           |   +---evsi/      event input files with event kinematics (Ntuples)
           |   +---fz/        event input/output files (fz)
           |   +---kine/      standard cards with particle properties definitions
           |   +---rz/        output histograms and Ntuples
           |   +---scan/
           |   +---wrld/
           +---bin/           examples of FFREAD cards for D0gstar run
           |   +---openvms_v6_1/  executables for different platforms
           |   +---sgi_53/
           |   +---sun4x_55/
           |   ...
           +---lib/
           |   +---openvms_v6_1/  libraries for different platforms
           |   +---sgi_53/
           |   ...
```

- **gene** — D0 standard physics generation steering package **d0gen**

[1] The standard DØ CVS repository is cvsuserd0chb.fnal.gov:/cvsroot/d0cvs which can be defined by setup d0cvs

- **d0si** — general DØ Simulation steering
  high level steering routines for event simulation
  high level steering for event reconstruction
  steering for kinematics
  steering for (full tracking) simulation
  steering for (re)digitization pass

- **detc** — Detector Constants — geometry related routines:
  general utilities for geometry, rotation matrices, materials, tracking media
  General Data cards interpretation, handling of GEAN and SETS data cards, etc...

- **user** — User 'hook' routines, mostly dummies by default

- **util** — general Utilities:
  Kinematics utility routines, simple generators and interfaces
  Utility routines common to simulation and reconstruction of simulated data
  Magnetic field routines
  Routines for random access events i/o
  input/output file handling package (IODB)
  Utilities for preparation of absorption and radiation lengths plots, etc...
  D0gstar versions of some GEANT routine

- **trak** — routines for digitization and hit reconstruction in the central Tracker ("TR")

- **calo** — ditto in the Calorimeter ("CA")

- **muon** — ditto in the Muon system ("MU")

- **reco** — steering and utilities for Reconstruction ("RE"):
  High level steering routines for event reconstruction
  Fanout and auxiliary routines for event reconstruction
  Utility routines for reconstruction
  User entry points in reconstruction, etc...

- **gcal** — the DØ version of GCALor[6].

1.2.3 DØ Unix environment

Upon login the variable pointing to the top level **D0gstar** directory should be set

```
setup d0gstar
```

to one of the machines group login scripts (d0dev, d0new, d0pro or d0old) are executed and several **D0gstar** environment variables are set:

$d0_path	– root directory of the public area
$d0_level	– level: old, pro, new or dev
$d0_ver	– version v000, v1_0, v1_0, v1_0
$d0_root	– root directory for the given version
$d0_src	– source directory for the given version
$d0_db	– data base directory
$d0_rz	– directory rz-files with histograms and Ntuples
$d0_fz	– directory with fz-files with events
$d0_evsi	– directory with Ntuples which defines kinematics
$d0_lib	– directory with compiled libraries for given architecture
$d0_libpath	– library path used in linking
$d0_bin	– standard directory with control cards to run **D0gstar**
$d0_exe	– directory which contains executables for given architecture
$d0_log	– directory log-files

The definition of some of these environment variables depends on the **D0gstar** version old/pro/new/dev being active. The user can change the **D0gstar** version via the following four commands:

- d0old → v000
- d0pro → v1_0
- d0new → v1_0
- d0dev → v1_0

the source code, and the corresponding compiled libraries and executables are kept in the above directories.

If after execution one of the above scripts (d0dev, d0new, d0pro or d0old) you don't get a message like

```
DOGSTAR version v1_0 has been initialized
```

```
with /tmp_root/1300/d0gstar/dev/d0gstar/bin/sgi_53
```

it means that you don't have the $D0GSTAR_DIR variable defined. You should defined it as

```
switch ( 'uname -n' )
         case 'd0chb':
  setenv DOGSTAR_DIR /tmp_mnt/tmp_root/1300/d0gstar
breaksw
default:
  setenv DOGSTAR_DIR /tmp_root/1300/d0gstar
breaksw
endsw
```

and

```
  alias d0dev "source $DOGSTAR_DIR/d0gstar/init/d0dev"
  alias d0new "source $DOGSTAR_DIR/d0gstar/init/d0new"
  alias d0pro "source $DOGSTAR_DIR/d0gstar/init/d0pro"
  alias d0old "source $DOGSTAR_DIR/d0gstar/init/d0old"
```

and repeate initialization.

1.2.4 DØ VMS environment

subs.vms On VMS as much as we can we try to keep one to one correspondence to **Unix** environment. Upon login to VMS mentioned machines group login scripts

```
$ d0_init  :=="@dafm02$dka300:[tmp1300.d0gstar.v1_0.d0gstar.init]d0_init.com"
$ d0dev    :=="d0_init dev"
$ d0new    :=="d0_init new"
$ d0pro    :=="d0_init pro"
$ d0old    :=="d0_init old"
```

are executed and several **D0gstar** logical names are set:

d0_path	– root directory of the public area
d0_level	– level: old, pro, new or dev
d0_ver	– version v000, v1_0, v1_0, v1_0
d0_root	– root directory for the given version
d0_src	– source directory for the given version
d0_db	– data base directory
d0_rz	– directory rz-files with histograms and Ntuples
d0_fz	– directory with fz-files with events
d0_evsi	– directory with Ntuples which defines kinematics
d0_lib	– directory with compiled libraries for Open VMS
d0_bin	– standard directory with control cards to run **D0gstar**
d0_sys	– given architecture
d0_exe	– directory which contains executables for given architecture
d0_log	– directory log-files
d0_cfl	– command file to compile and create libraries
d0_vax	– top directory with VMS sources

1.2.5 DØ data base

The **D0gstar** package uses a **Unix** directory/file based data base system. The data base files are accessed by a special package called **iodb**. The iodb code resides in the `util/iodb` directory.

There are utility routines for opening and closing the files, defining logical units etc. The unit numbers are defined implicitly so that the user need not worry about them. All the standard data files needed in the **D0gstar** simulation are given as defaults in fortran so that normally no system code (like link or assign) is needed in the job.

Instead of default files the user can define his/her own i/o files by using the FFREAD control cards CDIR and/or CFIL..

```
cdir 'EVTO' '../d0db/fz/neu_b '
cfil 'DETI' '../d0db/detc/geom_ini.rz '
     'EVSI' '../d0db/evsi/dtu2000evd0.ntpl '
     'EVTO' '../d0db/fz/d0_neu_c.fz '
     'HBKO' '../d0db/rz/d0_new_c.rz '
```

These cards (see section 1.5) are interpreted by the iodb package. The data base system is described in more detail in the reference manual section 2.3.3.

1.2.6 Example job scripts

A standard way of working (on d0sgi's) with the simulation package comprises the following steps:

• define pass to **D0gstar** top level directory (see section 1.2.3 • setup d0cvs, gtools • setup **D0gstar** variables – d0dev • create directory for a version of the **D0gstar**

```
mkdir v1_0
cd v1_0
```

• checkout from CVS repository (if it is necessary) and execute **$d0_mgr/d0boot**

```
cvs checkout d0gstar
```

or

```
mkdir d0gstar
cvs checkout d0gstar/mgr
cvs checkout d0gstar/src
cvs checkout d0gstar/d0db
```

• or just copy source files from $d0_src and/or from $d0_db

where anybody can find the latest version of codes and titles

• edit source

• compile with gmake (in d0gstar/v1_0)

```
d0boot production   # create production libraries
d0boot debug        # create debug libraries
```

• link with **$d0_mgr/d0gstar**

```
d0gstar             # for interactive version
d0gstar -p          # for interactive version with gpaw++
d0gstar -b          # for batch version
d0gstar -g          # for interactive debug version
d0gstar -b -g       # for batch debug version
d0gen               # d0gen
d0gen  -g           # d0gen debug version
```

In d0gstar script the first of all it is complied all files with fortran codes (*.F and *.f) in the current directory. In the link path search is included the current directory, ./lib/d0_sys directory and $d0_lib. Executable module is created in directory ./bin/d0_sys if it exists otherwise the module is created in the current directory.

• run the job script (examples can be found in $d0_mgr/run*)

For each of these steps there are sample scripts in the public directory **$d0_mgr**. Notice that before accessing these files one should be sure that the **D0gstar** environment is properly set. One gets e.g. the version v1_0, if one has applied the command d0new.

Thus the scripts and a related control card file are as follows:

- d0boot – compilation on **Unix**
- d0gstar – linking on **Unix**
- D0gstar.com – linking on VMS
- run – batch version of the simulation job
- d0vXXX.ttl – control card file for job

The file d0vXXX.ttl (XXX stands for version number) contains the FFREAD control cards.

1.3 Naming conventions

The naming convention for the source code is as follows: the routines of the package should in general start with the same two initials as the package itself, e.g. /detc/dcgeom. There may be exceptions from this rule especially in case of some utility routines which are in the **util** file.

The naming conventions for **GEANT** volumes, materials and media are described below.

1.3.1 Materials and tracking media naming conventions

We specify that any access to material or tracking media should be done only via a name (character string) instead by a hard-coded number. In general the name of the material and its related tracking media are the same with only one exception. For specific tracking media related to a given subsystem a prefix (a character plus underscore) is added as follows:

```
T_    --  for the tracker,
C_    --  for the calorimeter, and
M_    --  for the muon system.
```

Furthermore, if the material with this name does not exist, for example "*T_Air*", then the standard material will be used ("Air").

There is one special tracking medium that has the prefix "*Thick_*." This tracking medium is used to denote "thick" or "simple" (i.e., lacking internal structure) shielding materials and iron yokes for the purpose of reducing computer time. In volumes composed of this kind of medium, kinematical cuts are increased inside a sub-volume which is reduced from the mother volume by $3 \times x_0$ in all dimensions.

1.3.2 GEANT volume naming convention

The rule for the initial letter holds especially for the geometry volumes:

```
D  --  for general \DO volumes
B  --  for the beam region
T  --  for general tracker volumes
S  --  for the Silicon tracker
C  --  for the Scintillator fiber tracker
U  --  for the barrel calorimeter
E  --  for the endcap calorimeter
M  --  for the muon system
Q  --  for the low beta quadrupoles
```

Also "B" and "U" are devoted to barrel and "F","E" for endcaps. Their place in volume names is not fixed.

1.4 Program structure and data flow

1.4.1 General

The **D0gstar** simulation system includes the following phases:

- **d0gen** – standard DØ event generation package
- **d0geo** – creates GEANT geometry RZ-file from titles
- **d0kin** – kinematics for **D0gstar**
- **d0sim** – the DØ GEANT tracking and hits simulation
- **d0raw** – DØ raw data simulation, performs event pile-up and hit digitization
- **d0rec** – the DØ event reconstruction
- **d0ana** – the DØ event analysis

Each program phase can be run independently provided the output file from the previous phase is available. A component is controlled by FFREAD cards. The data flow diagram is presented in Fig.1.4.1. It uses GEANT and support from the **CERN** library. It has the following features:

- The geometry, materials, rotation matrices, magnetic field, etc. are specified in the database (now just a GEANT RZ-file) which uses as input title data cards which are read by the TZ (ZEBRA Titles) package;
- All communication among modules is done with GEANT ZEBRA structures.

The result of this is an RZ-file which is used for simulation (**d0sim** /**d0raw**) or reconstruction (**d0rec**). **d0sim** fills the KINE structure, does tracking, and makes hits. **d0raw** performs pile-up of events from a given number of bunches and makes digitization. The structures are output using the FZ packages for later analysis (in **d0rec**) or in combinations with trigger events for simulation of pile-up effects. The following sections give specific details concerning the different parts of this system (**d0geo** , **d0sim** and **d0rec**).

The detailed program flow charts are presented in the Reference Manual.

1.4.2 Event generation – d0gen

The DØ detector simulation package follows the principle that the interface between the physics generators and GEANT kinematics goes via intermediate files which have a well defined format: column-wise Ntuples [8]. The Ntuple files are made by the DØ standard physics generation package **d0gen** which works in terms of the HEPEVT common where particle kinematics are stored as a **Column-Wise-Ntuple** (single precision):

```
*****************************************************************
* Ntuple ID = 100     Entries = 2000       HEPEVT
*****************************************************************
* Var numb * Type * Packing *     Range      * Block   * Name     *
*****************************************************************
*       1   * I*4  *          *                * HEPEVT  * NEVHEP
*       2   * I*4  *          * [0,2000]       * HEPEVT  * NHEP
*       3   * I*4  *          *                * HEPEVT  * ISTHEP(NHEP)
*       4   * I*4  *          *                * HEPEVT  * IDHEP(NHEP)
*       5   * I*4  *          *                * HEPEVT  * JMOHEP(2,NHEP)
*       6   * I*4  *          *                * HEPEVT  * JDAHEP(2,NHEP)
*       7   * R*4  *          *                * HEPEVT  * PHEP(5,NHEP)
*       8   * R*4  *          *                * HEPEVT  * VHEP(4,NHEP)
*****************************************************************
*  Block    *  Entries  * Unpacked * Packed *   Packing Factor    *
*****************************************************************
* HEPEVT    *  2000     * 120008   * Var.   *    Variable         *
* Total     *    ---    * 120008   * Var.   *    Variable         *
*****************************************************************
* Blocks = 1            Variables = 8       Max. Columns = 30002 *
*****************************************************************
```

This provides the following benefits:

- files can be read on any system with **CERN** library installed
- in GUKINE only one interface code (HEPEVT→KINE) is needed
- events can be analysed with PAW, independent of GEANT

The HEPEVT standard is explained in ref. [9]. The event structure is in the common /HEPEVT/ shown below containing information on

```
      PARAMETER (NMXHEP=2000)
      REAL*4 PHEP, VHEP
      COMMON/HEPEVT/NEVHEP,NHEP,ISTHEP(NMXHEP),IDHEP(NMXHEP),
     &JMOHEP(2,NMXHEP),JDAHEP(2,NMXHEP),PHEP(5,NMXHEP),VHEP(4,NMXHEP)
```

particle status (ISTHEP), particle identifiers (IDHEP), mother-daughter relationships (JMOHEP, JDAHEP), particle 4-momenta (PHEP) and vertices (VHEP). For PHEP and VHEP we use single precision instead of PYTHIA's double precision.

HEPEVT is standard in PYTHIA. The STDHEP package [10] is used to interface the internal event structure of other physics generators with HEPEVT. The instruction how to build and run **d0gen** can be found in Section 2.

1.4.3 GEANT detector description – d0geo

The GEANT detector description consists of the files **detc** (detector description code), d0db/detc/*.tz (TZ titles for detector description and magnetic field maps). The main steering routine is DCGEOM (in **detc**) called from UGINIT (in **d0si**).

Details of the detector description code and data may be found in the reference manual.

1.4.4 Kinematics – d0kin

The event generator kinematics are available only via input Ntuples file. Run time generation (i.e. particle kinematics generation within **D0gstar**) is provided for simple events (IKINE=1-99) and Pythia events for **D0gstar** test purposes only .

For any other kind of generation, e.g. ISAJET generation at run-time, the user has to code his own USKIN9 routine which is called by GUKINE when IKINE=9.

The particle kinematics are controlled via the FFREAD control card "KINE" as follows:

```
 Kinematics definition:
================================================================================
IKINE = 0 ==>  Events are input from file (Ntuples)
        PKINE(1)    = not used
        PKINE(2-3)  = Etamin-Etamax
        PKINE(4-5)  = Phimin-Phimax (deg)
        PKINE(6-7)  = Ptmin -Ptmax  (GeV)
        PKINE(8)    = 1. if copy of HEPEVT wanted in VERT user bank
================================================================================
IKINE = 1 ==> Single particle event (call DOSIMPLE)
        PKINE(1)    = GEANT particle code
        PKINE(2-7)  = (as above)
================================================================================
IKINE = 2 ==> Simple MB event       (call DOSIMPLE)
        PKINE(1)   = (not used)
        PKINE(2-7) = (as above)
================================================================================
IKINE = 9 ==>  GUKINE calls user own taste generation USKINE
================================================================================
IKINE = 11 ==> Single particle event fixed momentum, Theta and Phi
        PKINE(1)  = GEANT particle code (def. muon+)
        PKINE(2)  = Theta (degree)      (def.   60.)
        PKINE(3)  = Phi (degree)        (def.   60.)
        PKINE(4)  = momentum (GeV/c)    (def.  100.)
        PKINE(5:6)= XYZ of primary vertex (def. 3*0)
        PKINE(9:10)= Sigma of X & Y of primary vertex (def. 3*0)
================================================================================
IKINE = 12 ==>  Multimuon event with fixed number of tracks
                in the given momentum and theta range
        PKINE(1)  = No. of muons        (def.    5)'
        PKINE(2)  = P_min (GeV/c)       (def.   1.)'
        PKINE(3)  = P_max (GeV/c)       (def.  50.)'
        PKINE(4)  = Theta_min (degree)  (def.  10.)'
        PKINE(5)  = Theta_max (degree)  (def. 170.)'
        PKINE(6:8)= XYZ of primary vertex (def. 3*0)'
        PKINE(9:10)= Sigma of X & Y of primary vertex (def. 3*0)'
================================================================================
IKINE = 13 ==>  Single track event in the given momentum and theta range
        PKINE(1)  = GEANT particle type (def. muon+)
        PKINE(2)  = p_min (GeV/c)       (def.   1.)
        PKINE(3)  = p_max (GeV/c)       (def.  50.)
        PKINE(4)  = Theta_min (degree)  (def.  10.)
        PKINE(5)  = Theta_max (degree)  (def. 170.)
        PKINE(6:8)= XYZ of primary vertex (def. 3*0)
        PKINE(9:10)= Sigma of X & Y of primary vertex (def. 3*0)
```

```
================================================================================
IKINE = 15 ==>  Single particle event, fixed transverse momentum, Theta and Phi
        PKINE(4)  = Transverse momentum (GeV/c)    (def.  100.)
        PKINE(1:3), PKINE(5:6) - the same as for IKINE = 11
================================================================================
IKINE = 16 ==>  Single track event in the given transverse momentum and theta range
        PKINE(2)  = p_T_min (GeV/c)     (def.    1.)
        PKINE(3)  = p_T_max (GeV/c)     (def.   50.)
        PKINE(1),PKINE(4:8)  are the same as for IKINE = 12
================================================================================
     IKINE = 17 ==>  Single track event in the given transverse momentum and eta range
        PKINE(2)  = p_T_min (GeV/c)     (def.    1.)
        PKINE(3)  = p_T_max (GeV/c)     (def.   50.)
        PKINE(4)  = eta_min              (no def.)
        PKINE(5)  = eta_max              (no def.)
        PKINE(6:8)          are the same as for IKINE = 12
================================================================================
IKINE >99 ==>  LUND event with MSEL = IKINE - 100 2 TeV proton-antiproton  interaction
        PKINE(1)  = p_T_min (GeV/c)  of parton-parton interaction
```

1.4.5 GEANT tracking and HITS generation – d0sim

The GEANT hits are stored in the GUSTEP routine which is called by the GEANT system at each step of the particle's tracing in the detector. Hit information is stored when the particle goes through a detector sensitive volume and this volume is defined as a detector element. The format of this information is defined in the Reference Manual. The muon and tracker hits have the same format. GUSTEP calls a user routine USSTEP which is dummy by default.

The control of what daughter particles (secondaries) will be stored in the KINE structure is also controlled in GUSTEP. Daughters are stored in KINE, if

1. the interaction takes place in the tracker cavity
2. and if the mother has $p_t >$ pTcut, or
3. the interaction takes place for muon with $p_t > 5 \times$ pTcut,
4. and if the interaction mechanisms are one of the following: decay, pair production, Bremsstrahlung or hadronic interaction.

The value of pTcut is 1.0 GeV. The tracker limits are defined by the constants Rcut and Zcut which are set to 74 cm and 139 cm, respectively.

1.4.6 Event pile-up, noise and digitization – d0raw

1.4.6.1 Event mixing and pile-up – gtrig

The event mixing with minimum bias (MB) background events is performed at the level of hits. "Physics" events and MB events are run separately event by event through **D0gstar** to produce HITS and intermediate files are output. The event mixing and pile-up are controlled by the following FF cards (see section 1.5).

Events containing the HITS structures are used in mixing using the CFIL card as follows:

```
CFIL  ...
      'FZIN'  'phevents.fz '      (input events must have HITS)
      'PILE'  'mbevents.fz '
or
      'EVSI'  'phevents.ntpl '    (GEANT tracking and HITS to be done)
      'PILE'  'mbevents.fz '
```

If an ntpl file (kinematics) is given as 'EVSI' input, one must have the card `KINE 0 ...` as well as SETS card set for the detector you want or use the SETS default (all detectors).

File names must be existing files or links pointing to existing files. Some sample files are in `$d0_db/fz` and `$d0_db/evsi` directories.

The number of mixed MB events per bunch is defined by the BEAM card and the number of bunches by the PILE card. With the PILE card one can define the number of bunch crossings before and after the crossing in which the physics event originated (the trigger crossing).

```
PILE  'Det1'  IF1  'Det2' IF2
```

The flag `IFi` is defined as

$$\mathrm{IFi} = 10N_{bc} + N_{ac}$$

where N_{bc} (N_{ac}) are the number of crossings before (after) the trigger crossing.
The defaults for pileup flags are -1 *which means that only the trigger crossing is simulated with no overlapping MB events.* For example the card

```
PILE  'TRAK'  10 'MUON' 31
```

would imply 5 crossings to be piled up for the muon detectors (3 before, the trigger and one after) and only 2 (one before and the trigger) to be piled up for the tracker detectors. Another example:

```
PILE  'TRAK'  0 'MUON' 1
```

would simulate only the trigger bunch for the Tracker with MB background and for muon system one additional bunch collision after the trigger (1 is equivalent with 01).

The information on tracks responsible for the hits is saved along with the hit elements for the trigger event. For superimposed minimum bias events, vertex and track information is not stored. Consequently the track numbers in the hit structure are set to 9000 + event number.

1.4.6.2 Digitization – gudigi

Conversion of the energy deposits to detectable signals and digitization are steered by the routine GUDIGI. The steering control is given by the FFREAD card DIGI. For example the card

```
DIGI  'TRAK'  1 'DO ' 0
```

will enable digitization in the the trackers whereas no digitization will take place for the rest of the detectors.

1.4.7 Reconstruction – d0rec

The event reconstruction is steered by the routine D0RECO in the **d0si** package. The reconstruction in subdetectors takes place in the following order:

1. Reconstruction in calorimeters,
2. Reconstruction in muon chambers,
3. Reconstruction in the Tracker and
4. Global reconstruction.

The steering control takes place by the FF control card RECO. The reconstruction result is stored in the RECO bank structure which is linked to the D0MT master bank. The reconstruction structure as well as the simulation structures are all under the same root D0MT as can be seen from the Appendix 'Bank Descriptions'. The top bank structure to support different sub-detectors reconstruction structures is organized as shown in the figure below.

1.4.8 Utilities

1.4.8.1 I/O file handling

1.4.8.2 GEANT extensions

1.5 User control cards

1.5.1 General

The **GEANT** program uses the FFREAD package to transfer user data to the program. In addition to the standard **GEANT** data cards (see **GEANT** manual BASE040) there are a few specific data cards defined for the **D0gstar** program. The data cards are either inserted in the job script file (batch) or read from a separate ASCII file (interactive job).

The data cards which control sub-detector specific operations are normally of the following form:

```
FFID 'KEY1' N1  'KEY2' N2  ...
```

where the keywords KEYi are normally subdetector identifiers and the numerical flags Ni are normally used as switch off/on (0/1) indicators, detector options or debug levels. The contents of this type of data card can be examined in the program by using the function ICFLAG as follows:

```
Nx = ICFLAG ('KEYx','FFID')
```

where 'FFID' (in) is one of the defined FFREAD keys, 'KEYx' (in) is the (detector) identifier and Nx (out) is the numerical flag value.

Specifications of GEANT defined control cards can be found from the GEANT manual, section BASE040. Various DØ specific data cards are explained below together with some of the most common GEANT data cards.

1.5.2 Printing, histograming and drawing control cards

The print/debug event range, the specific systems to be debugged and GEANT structures to be printed can be controlled by the following data cards:

```
DEBU  IFIRST ILAST IDELTA                 (Events to debug/print)
DBUG  'KEY1' N1  'KEY2' N2  ...           (Subsystems and levels to debug)
HBOOK 'KEY1' N1  'KEY2' N2  ...           (Subsystems to invoke HBOOK)
PRIN  'BNK1' 'BNK2' ...                   (Structures to print)
SWIT   N1 N2 N3 N4 ... N10                (Switches in interact. job)
```

Notice that the DEBU parameters controls over the DBUG, PRIN and SWIT flags and they become ineffective outside the event range determined by the DEBU card. The DBUG card controls subdetector specific debugs. For instance the control card

```
DBUG  'TRAK' 3  'MUON' 0
```

sets the tracker debug level to 3 and no debug from the muon system.

Only the first four characters of the FF-keys are significant so that HBOO and HBOOK are equivalent on input card. In the code only 4 characters should be used eg:

```
IVAL = ICFLAG ('TRAK','HBOO')
```

The HBOOK FF-card is used to control histograms and plots as well as Ntuples.

The DBUG and HBOOK data cards are designed to have some structure in the numerical flag values Nx. Nx is an eight bits and each of its bit specifies the flag value at a given stage of the program. The eight stages are defined as

1. Program initialization (INIT, **d0geo**)
2. Creation of the event kinematics (KINE, **d0kin**)
3. Tracking and recoring of hits (HITS, **d0sim**)
4. Event mixing and pile-up (PILE, **d0raw**)
5. Adding "junk" hits due to, for examples, neutron background (JUNK, **d0raw**)
6. Digitization and trigger control (DIGI, **d0raw**)
7. Creation of reconstruction hits (RHIT, **d0rec**)
8. Reconstruction (RECO, **d0rec**)

and they correspond to the respective bits (starting from the least significant bit).

In interactive GEANT the SWITches N1, ... , N4 are used as follows: A GEANT routine GDEBUG is called from GUSTEP and GDEBUG calls specific Printing/Storing/Drawing routines according to the SWIT flag values as described below:

```
GDEBUG calls  GPKIN            if  SWIT(1) = 2
              GSXYZ                SWIT(2) = 1   or   SWIT(3) = 1
              GPCXYZ               SWIT(2) = 2
              GDCXYZ (all)         SWIT(2) = 3
              GDCXYZ (charged)     SWIT(2) = 3   and  SWIT(4) = 3
              GSXYZ+GDTRAK         SWIT(2) = 4
```

The routines GP... print the track parameters (GPKIN) or points when tracking. The routines GD... draw points on graphics window so that the tracks are seen as lines. In interactive session one can change the SWIT parameters by SWITCH command (see GEANT XINT002–29).

1.5.3 Events and structures i/o control control cards

The following cards are used to control the i/o files and structures to be saved or read:

```
CDIR 'KEYW' 'directory '
CFIL 'KEYW' 'file '                    (space is significant)
SAVE 'KEY1' 'KEY2' ...
RSAV 'KEY1' 'KEY2' ...
GET  'KEY1' 'KEY2' ...
RGET 'KEY1' 'KEY2' ...
```

CDIR and CFIL cards are used to provide other than default DB files for the program.

The cards SAVE, RSAV, GET and RGET are standard GEANT control cards and they are used to control the structures (events) to be input/output in FZ or RZ format. As arguments they take the standard GEANT structure names like 'KINE', 'VOLU', 'HITS' and 'DIGI'. Also non-standard GEANT structure identifiers like 'RHIT' and 'RECO' can be given as arguments. The keyword 'INIT' has a special meaning: one can save and retrieve detector description files.

The control cards GET and RGET also affect the program flow: If a structure is requested from the input file, the processing request by the corresponding program control card will be overridden (see below).

1.5.4 Program flow control cards

The program flow control cards are listed below.

```
KINE  IKINE   PKINE(1)  PKINE(2)  ...  PKINE(8)
GEOM  'KEY1' N1  'KEY2'  N2        ...            (Detector)
SETS  'KEY1' N1  'KEY2'  N2        ...            (Hits)
PILE  'KEY1' N1  'KEY2'  N2        ...            (Pile-up)
DIGI  'KEY1' N1  'KEY2'  N2        ...            (Digitization)
JUNK  'KEY1' N1  'KEY2'  N2        ...            (Incoherent noise)
TRGP  'KEY1' N1  'KEY2'  N2        ...            (Trigger primitives)
RHIT  'KEY1' N1  'KEY2'  N2        ...            (Hit reconstruction)
RECO  'KEY1' N1  'KEY2'  N2        ...            (Reconstruction)
```

If a card is missing, default values are set for each keyword. Notice that the request by the above control card will be overridden if it is in contradiction with the GET/RGET card (see above).

The KINE card parameters go into the GEANT common /GCKINE/. It is used to control the kinematics input to GEANT. The meaning of the parameters is explained in section 1.4.4.

Keywords in the above list are normally *subdetector* volume names (see section 2.4.5.1). The value associated to the keyword is propagated to all lower levels and after that for all options except 'GEOM' the value for keyword is set equal the maximum value in the branch defined by this keyword.

The GEOM card is used to select geometry options. The 'KEYi' selection index Ni is defined as follows:

```
Ni = ID + 10*IT
```

where IT is a technology index and ID is the level of detail. The default technology index is IT=0 (baseline proposal). The levels of detail are:

```
ID = 3 .. detailed geometry (default)
     2 .. simplified geometry
     1 .. average geometry (no sensitive material)
```

For example, the GEOM card with the following contents:

```
GEOM 'DO  ' 3 'UC  ' -1
```

defines the default detector except that the barrel calorimeter will be completely switched off.

Activation and deactivation of the different subsystems of the detector (by the default all subsystems are active, keyword = 1) can be made, for example:

```
SETS 'DO ' 0 'WMUS' 1 'FMUS' 1
```

which means that all detector will be non-sensitive except the barrel and endcap muon chambers. The non-sensitive detector option means that there will be no HITS simulation.

Digitization, creation of trigger primitives, hit reconstruction and event reconstruction is controlled by DIGI, TRGP, RHIT and RECO cards. The default values of the corresponding detector flags are = 1. The user can switch off the action in certain detector by setting the corresponding value to 0, e.g.:

```
DIGI 'TRAK' 0
```

The flags can be used also to choose between different digitization and reconstruction options. More information may be found in the Reference Manual.

1.5.5 Other control cards

```
BEAM   X Y Z   dX dY dZ  Lum  NMB CROTIM XSECTN
(beam position and spread, luminosity, NMB, time between bunch crossings and
                                        total inelastic cross section)
```

The luminosity is given in units of $10^{32}\mathrm{cm}^{-1}\mathrm{s}^{-1}$. The mean number of events per bunch is calculated from:

$$\langle N_{MB} \rangle = L\sigma_T(p\bar{p})\tau_B.$$

where L is the luminosity and τ_B (= CROTIM) is the bunch crossing interval (ns). By default the actual number is generated according to Poisson statistics with the above mean. One can change this default by the parameter NMB which, when set to a positive value, implies a fixed number of mixed MB events. The BEAM card defines the nominal $p\bar{p}$ beam position and spread at the interaction point (in cm) as well as the luminosity in units of $10^32/\mathrm{cm}^2/\mathrm{s}$. The default values are $X = Y = Z = 0$ and $dX = dY = 50./\sqrt{2}$, $dZ = 30$ and $Lum = 1.$. The last parameters are optional and can be used to fix the number of MB events per bunch (see section 1.4.6.1), bunch crossing time and inelastic cross section.

Listed below is an example of control cards.

```
LIST
C    Nominal collision point and  sigmas(cm) Lumi/10**32  NmBunch CROTIM XSECTN
BEAM    0.     0.     0.    0.0050 0.0050 30.0     2.0              0    132.  61.5
C    *.ntpl no use   etamin etamax Phimin Phimax  pTmin pTmax
KINE   0       0.        -2.4     2.4       0.    360.     1.     9999.
TIME  5.    5.  1
RUNG  1    1                                (User run and first event number)
GEOM    'DO ' 3   'CALO' 1    'MUON' 3     (Geometry options)
SETS    'DO ' 1   'CALO' 0    'HCAL' 1     (HITS activation-deactivation)
CFIL   'HBKO' 'geant_hbook.hbook '          (Hbook output file)
       'META' 'geant.ps '                   (Graphics output file)
       'EVTO' '$SCRATCH/minbias.fz '        (Events output file)
SAVE 'HEAD' 'HITS' 'KINE' 'VERT'            (Output structures selection)
PILE 'DO ' 1                                (Pile-up control)
DIGI 'DO ' 0                                (Digitization control)
RHIT 'DO ' 0                                (Hits reconstruction control)
DBUG 'DO ' 1   'TRAK' 3                     (specific debug levels)
DEBU  -1    100   1                         (first,last,frequency to debug)
PRIN 'KINE'                                 (Structures to print)
SWIT    0  3   0   0   6*0                  (To see charged & neutrals)
STOP
END
```

1.6 Histogramming

To avoid clashes between different subsystems in **D0gstar** the following scheme is introduced for histogram subdirectories:

- directory /GEANT — contains histograms related to the GEANT processing of the events such as
  - time per event
  - time per deposited energy
  - memory used for event
  - total number of long life tracks per event
  - maximum stack size per event
  - ...
- directory /GENE — contains information about the generated (input) event: total and charged particle multiplicities, Z-coordinate of interaction, average energy versus pseudo-rapidity, p_T-distribution, ...
- directory /USER — contains histogram for the user

A directory structure closely following the structure defined in the SETUP title block (see section 4.5.1) is reproduced for the detector specific part. For example, the following subdirectories would be created:

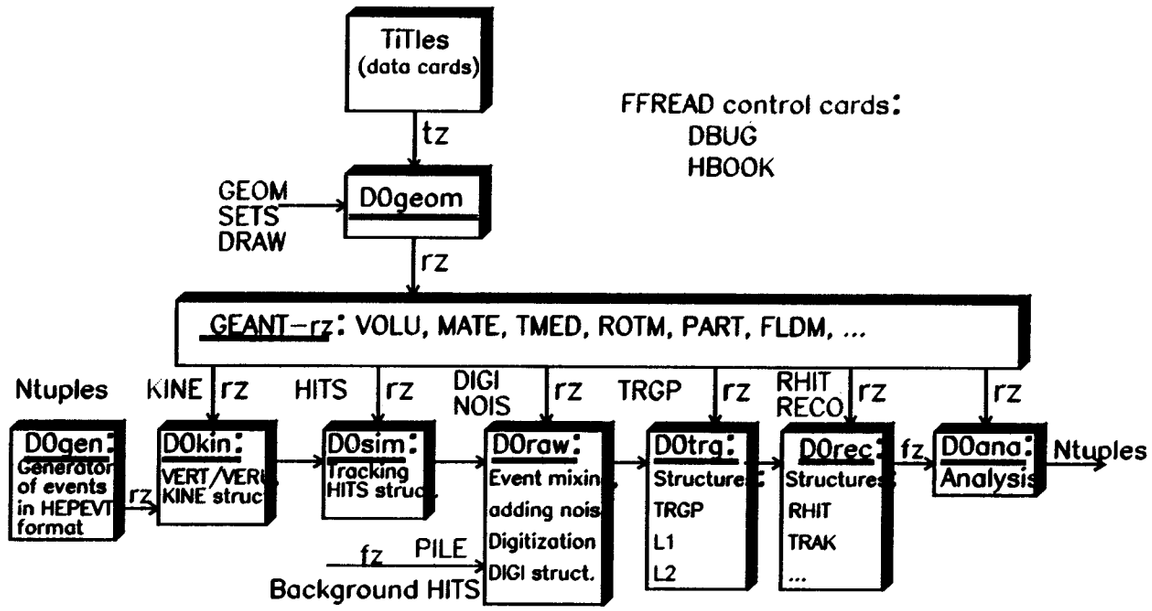
- directory /TRAK — contains histogram for the Central tracker
- directory /CALO — contains histogram for the calorimeter
- directory /MUON — ditto for the Muon system
- directory /TRIG — ditto for the trigger

1.7 User hooks

The user can interface with **D0gstar** on a different level via a set of "USER HOOKS." The list of these hooks is presented below:

- USS_FFKEY — FFREAD control cards and other beginning initializations
- US_INITG — initializations after reading FF cards
- US_INITH — user definition of histograms
- US_INITU — user definition of Ntuples
- USKINE — user definition of kinematics called when IKINE 3 (see control card "KINE").
- USSTEP — user action in GUSTEP
- US_OUT — user action prior to event i/o
- USLAST — user termination.

D0 GEANT based simulation framework



Part 2

Reference Manual

Chapter 2

Event generation - d0kin

The DØ simulation code, **D0gstar** , needs access to the particle kinematics produced from a number of distinct Monte Carlo generator programs. In general such programs have different ways of specifying the input parameters and a multitude of output event and file formats. The purpose of **d0gen** is to provide uniform input and output interfaces to the most common generators used by **D0gstar** . So far the only generator implemented within **d0gen** is Pythia.

To be consistent with **D0gstar** and CERNLIB we have adopted FFREADas the mechanism for specifying the input parameters to event generator programs.

Many event generators permit the conversion of the internal event format to the HEPEVT format.Moreover, the STDHEP packageprovides conversion routines to and from HEPEVT format for the Herwig, Jetset, Isajet, and QQ event generators. We have therefore adopted HEPEVT as the standard generator event format for **D0gstar** . **d0gen** is also a code repository of general event generator utilities which operate on the HEPEVT structure.

The output event file format adopted is the HBOOKcolumn-wise Ntuple format, which has a number of attractive features: it is system-independent and CERN-supported; it has a 1–1 mapping to Fortran COMMON blocks (*i.e.* HEPEVT); it is easy to read (by **D0gstar** , user programs, and PAW); and it supports transparently the packing of data.

2.1 d0gen code

The **d0gen** code is written in FORTRAN77. The source code may be found, on any machine with the DØ environment, in the CVS directory **d0gstar/src/gene**. While the **d0gen** code is part of the **D0gstar** software releases it is independent of the rest of the **D0gstar** code, apart from the use of standard IODB procedures (see **util/iodb**) to handle input/output files.

2.2 Creation of a HEPEVT Ntuple using d0gen

Event generation using **d0gen** has been implemented as follows:

1. the user modifies **GE_EVT_SEL** (if desired);
2. a standard script compiles, links to the **d0gen** library, and runs the program ;
3. the user controls the job with FFREAD data cards.

The following general data cards are understood by **d0gen** :

DEBUG IDEMIN IDEMAX ITEST

First event to debug, ast event to debug, Print control frequency — the same as used for **D0gstar**

`CFIL 'EVTO' 'higgs.ntpl '`
output HEPEVT event Ntuple file (N.B. a space after the file name and before the closing quotation mark is mandatory).

`KSEL i`
predefined DØ selection number `i` (`i=0` ⇒ call user selection routine `GES_USER`, the dummy version of which selects all events).

`ICUT i j k...`
INTEGER cut values (`KSEL`-dependent meaning).

`RCUT a b c...`
REAL cut values (`KSEL`-dependent meaning).

`TRIG n`
number of events to generate (e.g. `PYEVNT` calls).

`NSEL m`
number of events to be selected and written out.

`DECAY flag`
if true then **D0gstar** will provide decay for the particles known to it.

`Pythia flag frame beam target` $\sqrt{s}$
activate interface with Pythia (flag = true/flase)

`HERWIG flag beam target` p_{beam} p_{taget} `process` p_{Tmin}
activate interface with Herwig (flag = true/flase)

`ISAJET flag`
activate interface with ISAJET (flag = true/flase)

`LUGIVE flag`
card to force reading Pythia parameters from the input via GE_FFUSER

`ISABEG flag`
card to force reading ISAJET parameters from the input via GE_FFUSER
In order to change parameters via GE_FFUSER it should be used a pair of the control cards, for example,

```
*pythia true  'cms' 'p  ' 'p~  ' 2000.
lugive true
```

with "*" (star) in the first position of Pythia cards. During input it is used standard procedures of ISAJET (isaini) or Pythia (lugive) to read parameters for generators. Examples of job for different processes and with different generators are presented below:

- Generation of top production at Tevatron with Pythia The block of the Pythia parameters is started with card "LUGIVE" and ends with "END". The "*" in the first position and "!" mean comments.

```
list
DEBUG  -1   100   1
cfil 'EVT0' '../d0db/evsi/pythia_top.ntpl '
decay true
*pythia true  'cms' 'p  ' 'p~  ' 2000.
lugive true
*---
*--- parameter setting for jetset
*--- set strangness supression & fragmentation p_T
*---
*
* default settings
*
* LUDAT1         <**********************************
MSTJ(104)=6       ! allow top-antitop production
* LUDAT2         <**********************************
PMAS(C6,1)=175.   ! top quark mass
* LUDAT3         <**********************************
MDCY(C111,1)=0    ! inhibit pi0 decays
* PYPARS         <**********************************
MSTP(2)=2         ! (D=1) calculation of alpha_strong at hard interaction
PARP(31) = 1.2    ! specifies a k factor = 1.2 (SDC)
* force decay modes
*23456123451234512345  (6x,6i5)
DECAY    24  -13   14                    ! W --> mu+ nu
*
CKIN(1)=30.
CKIN(3)=2.
MSEL=6             ! Top pair production
END
luexec true (force decays and remove comment lines)
ksel 0 (user selection)
nsel 10000
trig 10
stop
```

- Generation of hard hadron process with Herwig

```
herwig true 'P' 'PBAR' 900. 900. 1500
luexec true
ksel 0 (user selection)
nsel 10000
trig 10
stop
```

- WW production job with ISAJET

```
*isajet true
isabeg
SAMPLE WPAIR JOB
1800,100,1,1/
WPAIR
BEAMS
'P','AP'
PT
50,100,50,100/
JETTYPE1
'W+','W-','ZO'/
JETTYPE2
'W+','W-','ZO'/
WMODE1
'E+','E-','NUS'/
WMODE2
'QUARKS'/
END
nsel 10000
trig 10
stop
```

2.3 Reading of HEPEVT Ntuples

Since the HEPEVT output events are in standard column-wise Ntuple format they can be used within PAW. They constitute the standard input format to **D0gstar** and can also be read from a standalone fortran program.

Chapter 3

Steering code — d0si

3.1 General

As mentioned in section 1.1.4.1, there are six logical units in the DØ simulation software which have common initialization and termination stages:

- **d0geo** — creation of GEANT-RZ file which contains as GEANT structures all detector constants including geometry, materials, tracking media, magnetic field and subdetector parameters. The **util** - and **detc** -packages (see below) provide utilities for this operation.
- **d0kin** — read from Ntuple or create event kinematics and convert it into GEANT KINE/VERT structure

- **d0sim** — the DØ detector simulation program which uses the GEANT structures provided by **d0geo** to
  - read or generate KINE structures for a given event
  - track all particles from this event through the detector elements
  - simulate detector hits
  - convert detector hits into digitization (i.e., the expected detector response).
- **d0raw** — the simulation of detector response
  - add hits from pile-up events
  - add "junk" – uncorrelated hits due to "neutron" background
  - convert detector hits into digitization (i.e., the expected detector response).
- **d0rec** — reconstruction of the event, including:
  - reconstruction of hits and showers from digitization
  - charged track pattern recognition
  - electromagnetic and hadronic shower reconstruction
  - ...
- **d0ana** — analysis of results

3.2 Initialization – uginit

The initialization for the batch version of the program is presented below[1]. This initialization stage program initializes ZEBRA and GEANT, sets and reads control cards (via FFREAD routines) and identifies the present versions of program and titles.

```
*
*   (*)  means  user code (not GEANT3 code)
*
* M A I N (*)   DO_GEANT3 Main Program
*  |
#if  defiend(DO_TRACE)
*  |->HANDLER(*)        condition handler
#endif
*  |
#if (defined(DO_MOTIF))&&(!defined(DO_BATCH))
   |->GPAWPP
#endif
#if (!defined(DO_MOTIF)&&(!defined(DO_BATCH))
   |->GPAW
#endif
#if  defined(DO_BATCH)
*  |->G Z E B R A       initialization of Zebra system
*  |->H L I M I T
#endif
```

[1](*) means user code (not GEANT3 code)

```
*  |
*  |->U G I N I T (*)  user initialization routine for DO_GEANT3
*  |    |                                                              |
*  |    |                    ====================                      |
*  |    |==================| Initialization phase|=====================|
*  |    |                    ====================                      |
*  |    |                                                              |
*  |    |->GINIT             GEANT initialization routine              |
*  |    |                    defines Simulation specific user data cards --|
*  |    |->FFSET                                                       |
*  |    |->DO_INITF (*)        General DO initializations: FFKEYs etc..    u
*  |    |    |->IO_FFKEY (*)  Define data cards used for I/O and database  n
*  |    |    |->DC_FFKEY (*)  ...                    for DETC              r
*  |    |    |->DO_FFKEY (*)  ...                    for DOSI              .
*  |    |    |->TR_FFKEY (*)  Define user data cards for tracker,          a
*  |    |    |->CA_FFKEY (*)  ...                                          d
*  |    |    |->MU_FFKEY (*)  ...                                          .
*  |    |    |->US_FFKEY (*)  User specific                                |
*  |    |->GFFGO             Read FF data cards                          |
*  |    |->GZINIT            Initialize GEANT data structures            |
*  |    |->DO_INITG (*)      DO initializations after reading FF cards   |
*  |    |    |->IO_INIT (*)  Initialize directories/files for I/O, database|
*  |    |    |->IO_OPEN_FILE (*)                                         |
*  |    |    |->UZINIT (*) Initialize user link area                     |
*  |    |    |->UTSELE (*) Interpretation of event selection FF card     |
*  |    |    |->TR_INITG (*)  Tracker Initialization                     |
*  |    |    |->CA_INITG (*)            ...                              |
*  |    |    |->MU_INITG (*)            ...                              |
*  |    |    |->US_INITG (*)  User Initialization                        |
```

3.3 DØ geometry preparation

The **d0geo** flow diagram is presented below. This part interfaces with UGINIT via a call to **DCGEOM** and tries to read the constants structure from the input GEANT.RZ file (using IODB key word "DETI" and "RGET" as a control card). If the requested file does not exist or there are any problems with the GEANT structure then the program recreates this structure using title data file(s) and saves them (structures) optionally as the output GEANT.RZ file (using IODB keyword "DETO" and "RSAV" as the control card).

```
*  |    |              ============================                  |
*  |    |  ============|Data Base creation/reading|=================|
*  |    |              ============================                  |
*  |    |->DCGEOM   (*)  steering for reading of detector titles      |
*  |    |    |->DC_PINI (*)    particle table initialization          |
*  |    |    |    |->PYINIT           Use PYTHIA particle tables       |
*  |    |    |    |->LUGIVE           Use JETSET interpreter           |
*  |    |    |->IO_OPEN_FILE (*) Opens the input rz file              |
*  |    |    |->TZINIT    Read titles, initialize field map, creates  |
*  |    |    |    |        DO geometrical setup                       |
*  |    |    |    |->TZUSER (*)  Interprets the title banks and converts |
*  |    |    |         |          into GEANT structure                 |
*  |    |    |         |->DCSETU (*)     Interpret SETUP data card     |
*  |    |    |         |    |->DC_SETUP (*)                            |
*  |    |    |         |       |->DCDECO (*)                           |
```

```
*  |   |   |        |       |->DC_FORM (*)                                  |
*  |   |   |        |->DCONST (*)     Load material constants               |
*  |   |   |        |   |->DC_ACT (*)                                       |
*  |   |   |        |       |->DCGMAT (*)                                   |
*  |   |   |        |       |->DCGMIX (*)                                   |
*  |   |   |        |       |->DCTMED (*)                                   |
*  |   |   |        |       |->DCTPAR (*)                                   |
*  |   |   |        |       |->DCROTX(*)   rotation matrices                |
*  |   |   |        |       |->DCSENS(*)                                    |
*  |   |   |        |       |->DCMEDI(*)                                    |
*  |   |   |        |->DCVALU (*) Test validity of the title block          |
*  |   |   |        |->DCGEAN (*)                                           |
*  |   |   |        |   |->DCVALU (*)                                       |
*  |   |   |        |   |->CLTOU                                            |
*  |   |   |        |   |->DC_ACT (*)                                       |
*  |   |   |        |       |->DCGMAT (*)  Load material table              |
*  |   |   |        |       |->DCGMIX (*)  Load mixture table               |
*  |   |   |        |       |->DCTMED (*)  Load tracking medium table       |
*  |   |   |        |       |->DCTPAR (*)  Load special TMED parameters     |
*  |   |   |        |       |->DCCKOV (*)  Define constants for Cerenkov    |
*  |   |   |        |       |->DCROTX (*)  Define rotation matrices         |
*  |   |   |        |       |->DCSENS (*)  Declare sensitive medium         |
*  |   |   |        |       |->DCMEDI (*)       ...                         |
*  |   |   |        |       |->DCVOLU (*)  Create a new volume              |
*  |   |   |        |       |->DCDVN  (*)   Create division                 |
*  |   |   |        |       |->DCDVN2 (*)       ...                         |
*  |   |   |        |       |->DCDVT  (*)       ...                         |
*  |   |   |        |       |->DCDVT2 (*)       ...                         |
*  |   |   |        |       |->DCDVX  (*)       ...                         |
*  |   |   |        |       |->DCNEAR (*)  Order the volumes                |
*  |   |   |        |       |->DCNEXT (*)       ...                         |
*  |   |   |        |       |->DCORD  (*)       ...                         |
*  |   |   |        |       |->DCPOS  (*)  Position the volumes             |
*  |   |   |        |       |->DCPOSP (*)       ...                         |
*  |   |   |        |       |->DCSATT (*)  Set attributes                   |
*  |   |   |        |       |->DCDET  (*)  Declare sensitive detector       |
*  |   |   |        |       |->DCDETV (*)       ...                         |
*  |   |   |        |       |->DCDETA (*)  Declare an alias                 |
*  |   |   |        |       |->DCDETH (*)  Define hit elements              |
*  |   |   |        |       |->DCDETD (*)  Define digit elements            |
*  |   |   |        |       |->DCDETT (*)  Define trigger primitives        |
*  |   |   |        |       |->DCDETU (*)  Define user words                |
*  |   |   |        |       |->DCHITS (*)  Standard hit banks               |
*  |   |   |        |       |->DCHITR (*)  Standard RHIT banks              |
*  |   |   |        |       |->DCPGON (*)                                   |
*  |   |   |        |       |->DCTPNT (*)  Position two-point volumes       |
*  |   |   |        |->DCDRAW (*)     Interface to GDRAW routines           |
*  |   |   |        |->GGCLOS                                               |
*  |   |   |->IO_CLOSE_FILE (*)                                            |
*  |   |   |->DC_SATT (*)         Set special attributes                   |
*  |   |   |->DCPART (*)         Complete particle table or                |
*  |   |   |->DCPARX (*)         Update particle table in memory           |
*  |   |   |->GGCLOS                                                       |
*  |   |   |->DCMEDI (*)         Special action for tracking media         |
*  |   |   |   |->GPHYSI                                                   |
```

```
*  |    |    |->GLOOK                  Print GEANT structures                |
*  |    |    |->GPMATE/GPTMED/GPROTM/GPVOLU/GPSETS/GPPART                    |
*  |    |    |->DC_PRNT (*)            Print run conditions                  |
*  |    |    |                                                               |
```

Initialization of histograms and Ntuples.

```
*  |    | ============  Initialization of HBOOK ======================== |
*  |    |                                                                  |
*  |    |->DO_INITH (*)                                                    |
*  |    |    |->HTITLE                                                     |
*  |    |    |->IO_OPEN_FILE (*) Open hbook file if any                    |
*  |    |    |->DO_INITU (*)     Initialize Ntuples                        |
*  |    |    |    |->TRE_INITU (*)  for Tracker                            |
*  |    |    |    |->CA_INITU  (*)  for CALO                               |
*  |    |    |    |->US_INITU  (*)  for Muon system                        |
*  |    |    |    |->RE_INITU  (*)  for reconstruction                     |
*  |    |    |    |->US_INITU  (*)  for user                               |
*  |    |    |->HMDIR                                                      |
*  |    |    |->G_INITH (*) book GEANT standard histograms                 |
*  |    |    |->TR_INITH (*) initialize subsystem  histograms              |
*  |    |    |->CA_INITH (*)                                               |
*  |    |    |->MU_INITH (*)                                               |
*  |    |    |->US_INITH (*)                                               |
*  |    | ============ Initialization of input/output fz-files  =========|
*  |    |                                                                  |
*  |    |->IO_OPEN_FILE (*)     Open fz-file for input/output:             |
*  |    |->ZPHASE(1)         Flag end of initialization stage              |
*  |                                                                       |
```

3.4 Simulation steering

Simulation steering is provided for

- at the level of the run — definition via FFREAD-cards of the run parameters, kinematics, cuts, level of simulation (hits and digitization), choice and initialization of input/output files fz-files for trigger event, pile-up events and output of the simulation results; initialization of HBOOK Ntuples and histograms (see section about creation of histograms)
- at the level of the event — event reading, selection, processing and writing, execution error trapping
- at the level of the step — creation of hits, histograming. In this step there is also special treatment of particle decays (see the section describing kinematics and particle definitions).

and includes:

- initialization of input/output fz-files for trigger event, pile-up events and output of the simulation results
- initialization of HBOOK Ntuples and histograms. It is essential that at this stage different directories are created for different subsystems in order to avoid interference among them.

- loop over requested number of events, create hits and fill histograms.

```
*  |->Q N E X T E      loops over events                                    |
*  | (G R U N)                                                              |
*  |   |                                                                    |
*  |   |->TIMEX            gets time left after initialization              |
*  |   |                                                                    |
*  |   |<-------------------------------------------------------------------|
*  |   |                                                                    |
*  |   |->GTRIGI             initialization for event processing            |
*  |   |-> GTRIG (*)         event processing                               |
*  |   |   |->GSCANK/GUTREV/GSCANO  geantino SCAN if any                    |
*  |   |   |->TIMED                                                         |
*  |   |   |->GCFIN (*)      try to read information from FZ file           |
*  |   |   |->UTEVSL (*)     select event from FZ-file                      |
*  |   |   |                                                                |
*  |   |   |========== d0kin                                    ========|
*  |   |   |                                                                |
*  |   |   |->GUKINE (*)     generates or reads event input kinematics      |
*  |   |   |                                                                |
*  |   |   |========== d0sim                                    ========|
*  |   |   |                                                                |
*  |   |   |->GUTREV (*)     controls tracking of a complete event          |
*  |   |   |                                                                |
*  |   |   |========== d0raw       ======================================|
*  |   |   |========== pile-up     ======================================|
*  |   |   |->DO_PILE (*)                                                   |
*  |   |   |   |->GPOISS                                                    |
*  |   |   |   |->IO_OPEN_FILE(*)open file with pile-up events              |
*  |   |   |   |->GCFIN (*)      read Pile-up event hits if any             |
*  |   |   |   |->UTPRNT(*)                                                 |
*  |   |   |   |->IO_CLOSE_FILE (*)                                         |
*  |   |   |   |->GFHITS                                                    |
*  |   |   |   |->GSAHIT (*)                                                |
*  |   |   |========== junk        ======================================|
*  |   |   |->DO_JUNK   (*)                                                 |
*  |   |   |   |->TR_JUNK  (*)                                              |
*  |   |   |   |   |->TR_RATE (*)  rate versus Z and R                      |
*  |   |   |   |->MU_JUNK  (*)                                              |
*  |   |   |   |   |->M_RATE (*)  rate versus Z and R                       |
*  |   |   |   |->TIMED                                                     |
*  |   |   |   |->UTPRNT (*)                                                |
*  |   |   |->GRLEAS                                                        |
*  |   |   |========== digitization  ================================|
*  |   |   |->GUDIGI (*)     fan-out routine for digitization               |
*  |   |   |   |->DOHITL(*) get list of hitted detector elements            |
*  |   |   |   |->GFPARA    fetch detector element volume parameters        |
*  |   |   |   |->GFDETU    fetch detector element user parameters          |
*  |   |   |   |->GFPATH    get volume list from path                       |
*  |   |   |   |->GLVOLU    load the given volume position in /GLVOLU/      |
*  |   |   |   |->GFHITS    fetch hits                                      |
*  |   |   |   |->TSH2D (*) digitization for Silicon                        |
*  |   |   |   |->TCH2D (*)     -"-      for Sci.Fiber                      |
*  |   |   |   |->CAH2D (*)     -"-      Calorimeter                        |
*  |   |   |   |->MUH2D (*)     -"-      Muon system                        |
```

```
*  |    |    |    |->TIMED                                                    |
*  |    |    |   |========== Trigger primitives ==============================|
*  |    |    |   |->D0D2T (*)       fanout for trigger primitives            |
*  |    |    |    |   |->GFDETU                                              |
*  |    |    |    |   |->MUD2T (*)  Muon trigger primitives                  |
*  |    |    |   |->GPTRGP (*)  Print trigger primitives if any             |
*  |    |    |   |========== 1 Level Trigger    ==============================|
*  |    |    |   |->D0L1_TRG (*)      L1 Trigger reconstruction             |
*  |    |    |   |========== 2-nd Level trigger ==============================|
*  |    |    |   |->D0L2_TRG (*)                                            |
```

3.5 Kinematics — d0kin

3.5.1 Kinematics data structure

The GEANT KINE and VERT data structures are used for the track and vertex kinematics. These structures can be read in **D0gstar** as a column-wise Ntuple. The essential point is that all particles from Table1 above which decay via weak or electromagnetic interactions are declared stable at the event generation stage. All decays and secondary interactions are treated by GEANT. Thus the kinematics input information contains only one vertex and KINE-structure which corresponds to one primary proton-antiproton interaction. Information about the primary interaction is stored in HEP-format for each primary vertex in the user VERU-bank with cross references to the KINE structure.

During the processing of this event in **D0gstar** , new vertices are created corresponding to decays and secondary interactions of particles. The GEANT protocol provides the complete history of each particle generated in any step. The criteria under which secondary particles are selected for storage into a VERT/KINE structure are the following:

- the interaction takes place in the tracker cavity
- and if the mother has $p_t > 1$ GeV/c, or
- the interaction takes place for muon with $p_t > 5$ GeV/c,
- and if the interaction mechanisms are one of the following: decay, pair production, Bremsstrahlung or hadronic interaction.

For tracker studies it is important to store the V^0 and flavour decay secondaries and for muon decay background studies it is important to store all the $K \rightarrow$ **muon** and $\pi \rightarrow$ **muon** decays which would potentially reach the muon system.

3.5.2 Vertex fluctuation

The primary vertex nominal position and sigmas for fluctuation are defined by the FFREAD card BEAM which is defined on page 1.5.5. The routine VTXGEN is used for the vertex fluctuation.

Fixed vertex position can be obtained by setting the BEAM card parameters 4,5 and 6 to 0. 0. 0. which implies no vertex fluctuation.

The standard GUKINE flow chart looks as follows:

```
*  |   |   |========== d0kin                                          ========|
*  |   |   |->GUKINE (*)     generates or reads event input kinematics      |
*  |   |   |   |                                                            |
*  |   |   |   |->DONTUP (*) kinematics read from Ntuples                   |
*  |   |   |   |   |->DONT_READ (*)                                         |
*  |   |   |   |   |   |->DO_HEPC (*)                                       |
*  |   |   |   |   |   |->LULIST                                            |
*  |   |   |   |   |->DO_HEPG (*)                                           |
*  |   |   |   |       |->VTXGEN (*)                                        |
*  |   |   |   |       |   |->GRANOR                                        |
*  |   |   |   |       |->GSVERT                                            |
*  |   |   |   |       |->DOPYTIDG (*)                                      |
*  |   |   |   |       |->GSKINE (*)                                        |
*  |   |   |   |->DOSIMPLE (*) single particle event with fixed theta,      |
*  |   |   |   |   |          phi, p_T                                      |
*  |   |   |   |   |->VTXGEN (*)                                            |
*  |   |   |   |   |->TRAMUL (*)                                            |
*  |   |   |   |   |->PARTYP (*)                                            |
*  |   |   |   |   |->GIVEPT (*)                                            |
*  |   |   |   |   |->GSVERT                                                |
*  |   |   |   |   |->GSKINE                                                |
*  |   |   |   |->USKINE (*)  user kinematics                               |
*  |   |   |   |   |->USKIN1 (*)                                            |
*  |   |   |   |   |->USKIN2 (*)                                            |
*  |   |   |   |   |->USKIN3 (*)                                            |
*  |   |   |   |   |->USKIN6 (*)                                            |
*  |   |   |   |   |->USKIN7 (*)                                            |
*  |   |   |   |   |->USKIN8 (*)                                            |
*  |   |   |   |   |->USKIN9 (*)                                            |
*  |   |   |   |   |->USLUND (*)                                            |
*  |   |   |   |   |->LULIST                                                |
*  |   |   |   |->GPVERT                                                    |
*  |   |   |   |->GPKINE                                                    |
```

For cuts defined by the KINE card one uses the integer function ICKREJ which takes the particle momentum as an input argument and returns 1, if the particle is outside the cuts.

3.6 GEANT tracking — d0sim

This section presents the steering of the GEANT tracking

```
*  |   |   |========== d0sim                                          ========|
*  |   |   |                                                                  |
*  |   |   |->GUTREV (*)     controls tracking of a complete event            |
*  |   |   |   |                                                              |
*  |   |   |   |->GTREVE                                                      |
*  |   |   |       |-> store initial tracks in stack                          |
*  |   |   |       |                      (-> GFKINE,GSSTAK,GUSKIP)           |
*  |   |   |       |                      ( loop over tracks )                |
*  |   |   |       |->GLTRAC   prepare commons for tracking                   |
*  |   |   |       |      | ->GMEDIA          finds current volume          |
*  |   |   |       |                                                          |
*  |   |   |       |->GUTRAK (*)       calls GTRACK                           |
```

```
*  |   |   |          |
*  |   |   |          |->GTRACK            controls tracking of current
*  |   |   |              |                 track
*  |   |   |              |
*  |   |   |              |->GTMEDI            finds current volume
*  |   |   |              |
*  |   |   |              |  GTGAMA or
*  |   |   |              |  GTELEC            extrapolation of particle
*  |   |   |              |  GTNEUT
*  |   |   |              |->GTHADR            track  by  one  step
*  |   |   |              |  GTNINO
*  |   |   |              |  GTMUON
*  |   |   |              |   |->GUDCAY (*)    decay of particles
*  |   |   |              |       |->LU1ENT
*  |   |   |              |       |->LUEXEC
*  |   |   |              |           |->LUDECY
*  |   |   |              |
*  |   |   |              |->GFINDS            Compute SET and DET number
*  |   |   |              |                    if volume is sensitive
*  |   |   |              |
*  |   |   |              |->GUSTEP (*)        special operations at the
*  |   |   |                  |                step
*  |   |   |                  |->GSCANU (*)
*  |   |   |                  |   |->GSSCAN (*)
*  |   |   |                  |->GDEBUG
*  |   |   |                  |->DOMTHIT (*)    stores hits
*  |   |   |                  |   |             for current detector
*  |   |   |                  |   |->GMTOD
*  |   |   |                  |   |->GSAHIT (*)
*  |   |   |                  |   |->UTPRNT (*)
*  |   |   |                  |->DOCHIT (*)
*  |   |   |                  |   |->GSCHIT (*)
*  |   |   |                  |->USSTEP (*)    user action on this step
*  |   |   |                  |->GSKING        Store any secondary
*  |   |   |                                   particles generated during
*  |   |   |                                   step
```

3.6.1 Reconstruction steering

This section presents the steering for muon track reconstruction.

```
*  |   |   |==========  d0reco                                    ========|
*  |   |   |==========  Hit reconstruction ==============================|
*  |   |   |->D0RHIT(*)          create RHIT structure                    |
*  |   |   |   |->GFATT                                                   |
*  |   |   |   |->GFPARA                                                  |
*  |   |   |   |->GFDETU                                                  |
*  |   |   |   |->D0D2R (*)                                               |
*  |   |   |   |   |->D0DIGL (*)                                          |
*  |   |   |   |   |->GFPATH                                              |
*  |   |   |   |   |->GLVOLU                                              |
*  |   |   |   |   |->GFDIGI                                              |
*  |   |   |   |   |->TSD2R (*) Si   reconstructed hits from digi.        |
*  |   |   |   |   |->TCD2R (*) SCFT    -"-                               |
```

```
*  |   |   |   |   |->CAD2R (*) CALO    -"-
*  |   |   |   |   |->MUD2R (*) MUON    -"-
*  |   |   |   |->D0H2R (*)   smeared hits
*  |   |   |   |   |->D0HITL (*)
*  |   |   |   |   |->GFPATH
*  |   |   |   |   |->GLVOLU
*  |   |   |   |   |->GFHITS
*  |   |   |   |   |->D0H2R1 (*)
*  |   |   |   |   |->GSRHIT
*  |   |   |   |->TIMED
*  |   |   |->GPRHIT
*  |   |   |========== Reconstruction ==================================|
*  |   |   |->DORECO (*)
*  |   |   |   |->REEFIT (*)    Find em clusters
*  |   |   |   |->REJFIT (*)    Find jets
*  |   |   |   |->REMFIT (*)    Muon Track fit
*  |   |   |   |   |->REGETM (*)  Read hits
*  |   |   |   |   |   |->REINIT (*)
*  |   |   |   |   |   |->GFDETU    Get detector parameters
*  |   |   |   |   |   |->D0HITL (*) Get list on hitted elements
*  |   |   |   |   |   |->GFPATH    Detector element geometrical
*  |   |   |   |   |   |->GLVOLU    parameters
*  |   |   |   |   |   |->GFRHIT (*) Get RHIT
*  |   |   |   |   |   |->GFHITS (*) Get HITS (for debug only)
*  |   |   |   |   |   |->REFMEA (*) transformation of RHITs  into
*  |   |   |   |   |   |       |        measurement structures
*  |   |   |   |   |   |       |->LGPRED (*) Parameters of measurement plane
*  |   |   |   |   |   |->RETSEG (*)     Find Track segments
*  |   |   |   |   |   |->REFNTK (*)     Find Tracks from Track segments
*  |   |   |   |   |   |->REFITK (*)     Track fit
*  |   |   |   |   |   |->RETPIP (*)     Transport track parameters in IP
*  |   |   |   |   |   |    |->RETGLB (*)
*  |   |   |   |   |   |->RETKID (*)
*  |   |   |   |   |   |->REPRNT (*)
*  |   |   |   |   |->RETFIT (*)
*  |   |   |   |       |->REGETM (*)
*  |   |   |   |       |->REFITK (*)
*  |   |   |   |       |->RETPIP (*)
*  |   |   |   |       |->RETKID (*)
*  |   |   |   |       |->REPRNT (*)
*  |   |   |   |       |->HFNTB
```

3.6.2 Event summary

```
*  |   |   |->GUOUT (*)
*  |   |   |   |->C_OUT (*)
*  |   |   |   |   |->HFNTB
*  |   |   |   |   |->GFVERT
*  |   |   |   |   |->GFATT
*  |   |   |   |   |->GFHITS
*  |   |   |   |->TR_OUT (*)
*  |   |   |   |->CA_OUT (*)
*  |   |   |   |->MU_OUT (*)
*  |   |   |   |->US_OUT (*)
```

```
*  |    |    |    |->GCFOUT (*)
*  |    |    |    |    |->FZOUT
*  |    |    |    |->UTPRNT (*)
*  |    |    |->DOFHIS (*)     Fill standard GEANT histograms
*  |    |    |->DOHSAV (*)     save histograms if needed
*  |    |
*  |    |->GTRIGC   Clear event partition
```

3.7 Termination phase

The termination phase is standard; histograms are saved and files closed.

```
*  |
*  |============= Run termination ======================================|
*  |
*  |->UGLAST (*)      User termination
*  |    |->USLAST(*)  User entry point
*  |    |->IO_CLOSE_FILE (*) Close IO Buffers
*  |    |    |->IO_FIL_KADDR (*)
*  |    |    |->IO_UNIT (*)
*  |    |    |    |->IO_FIL_KADDR (*)
*  |    |    |->IO_FILE (*)
*  |    |    |    |->IO_FIL_KADDR (*)
*  |    |    |->HREND
*  |    |    |->FZENDI
*  |    |    |->FZENDO
*  |    |    |->HROUT
*  |    |->DOHSAV (*)
*  |    |    |->PYSTAT
*  |    |    |->HPAKE
*  |    |    |->IO_UNIT (*)
*  |    |    |->IO_FILE (*)
*  |    |    |->HRFILE
*  |    |    |->HROUT
*  |    |    |->HREND
*  |    |->IO_UNIT (*)
*  |    |->HPLEND
*  |    |->GDCLOS
*  |    |->GLAST
*  |
* STOP ================================================================|
```

Chapter 4

Detector description code and constants — detc

4.1 General

The organization of the geometry, material and tracking media constants in **D0gstar** is based on the following principles:

1. All detector geometry related constants are defined in external files. TZ (ZEBRA Title package format) titles are used as initial storage for these constants. The detailed format of the title banks may be detector dependent.

2. The constants are considered first of all as instructions for direct filling of the corresponding GEANT structures — volumes, materials and tracking media, rotation matrices, detector elements definition and detector dependent information for description of the corresponding detector element response. At this stage title constants define objects corresponding to GEANT entities with parameters as attributes of these objects.

3. The geometry defined by these titles (GEAN banks) is built with the TZ utility routines in the **detc** source file. The experimental hall, master-volumes for all subsystems, as well as volumes not belonging to a specific subsystem (e.g., quadrupoles, shielding, etc) are defined in the same way.

4. The geometry definitions include information to allow for version selection.

5. There is a set of standard rotation matrices defined by the ROTM title bank with a 4-character keyword for each matrix. A standard rotation matrix is accessed by using this keyword in a call to a utility routine. This set includes as "NEXT" for any additional rotation matrix. For the additional rotation matrices an identifier number is given incremented by one from the largest number so far.

6. Material and tracking media are always referred to by their names.

This approach has been introduced and discussed in detail in a CMS technical note[12]. As has been pointed out above, in **D0gstar** all the experimental setup related code and constants (geometry, materials, rotations, hits-formats, magnetic field map, user-defined detector parameters, ...) are centralized into the three following sources (see flow diagram in section 3.3):

- The source file **detc** (GEANT Detector Constant Definition Language) with FORTRAN routines which contain all non-trivial parametrical models for all subsystem (detector) descriptions.

- The title files in d0db/detc (title information for the detector) which contains in the ZEBRA Title Package format (TZ-format) all constants and user parameters for the above models.

The non-trivial part is an interface between the titles data banks and the FORTRAN codes (from Detector Constant package) describing the model for a specific subdetector. To establish that we define title banks of two types:

- Title banks which contain parameters

- Title banks of specific formats for volume definitions which are interpreted by specific utility routines of the **detc** package. The style of format definitions which is used:

  - "GEAN" style contains a bunch of trivial GEANT instructions like creation of volume, its positioning, hit definition, detector set definition, setting an attribute to the volume, and so on.

  Only this type of title banks have assignment to keywords and version numbers which have been discussed in the version selection mechanism (see section 1.5).

As examples, shown in Fig.1, Fig.2 and Fig.3 are drawings of the DØ detector obtained in GEANT using the above title banks for Run I and Run II configurations and the calorimeter tower structure. The formats of the title banks will be described in detail below.

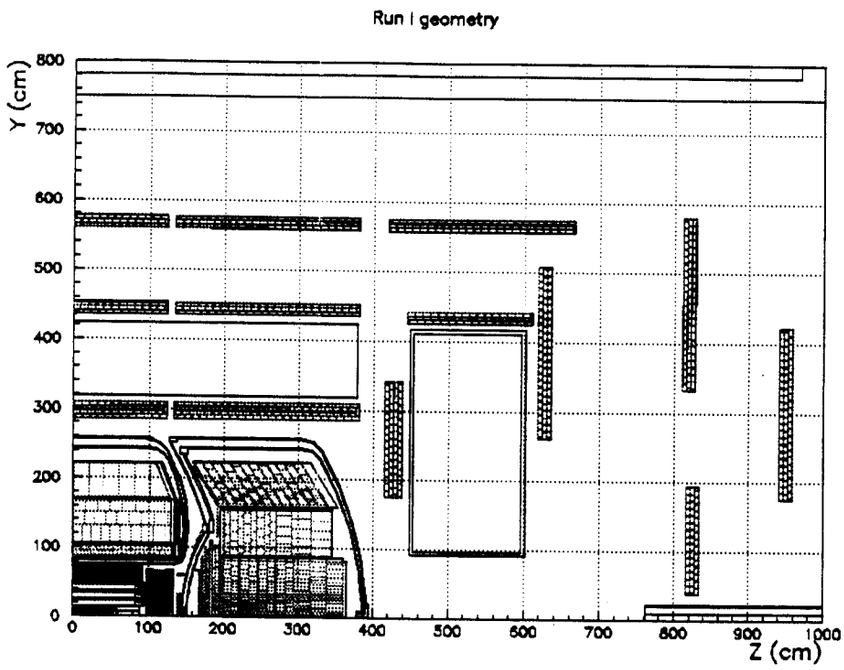


Figure 1: The DØ detector GEANT description for Run I.

4.2 Magnetic field

The field map(s) is placed in the d0db/detc/fmap.tz file in TZ format. The ZEBRA-bank with this table is kept along with other GEANT initialization structure and can be written to and read in from standard GEANT RZ-file. The field maps used for the linear interpolation are placed in the three following title banks for the **2D** solenoid magnetic field map inside the solenoid (FLDM) :

```
*--                                                  10/25/96  Ryuji Yamada
*-- D0 Upgrade Magnetic Field Map for Whole D0 Detector "d0fieldmap.grid"
*-- Geometry : Base    + 2T Solenoid     4643.705 x 1.0223 = 4747.26 A
*--                    + Toroidal Magnets  CF + both EF   1500 A
*--                                        both SAMUS        0 A
...
*D0  FLDM -IF -E15360        #. Internal solenoid 2D-field map
   #. NR, NZ, DR, DZ, Rmin, Rmax , Zmin, Zmax, B_rho(NR*NZ),*B_z(NR*NZ
     76. 101. 2.0 2.0   0.0 150.0    0.0 200.0
#. B_r
      44*  0.000        6* -0.001       29*  0.000        4* -0.001        6* -0.002
...
```